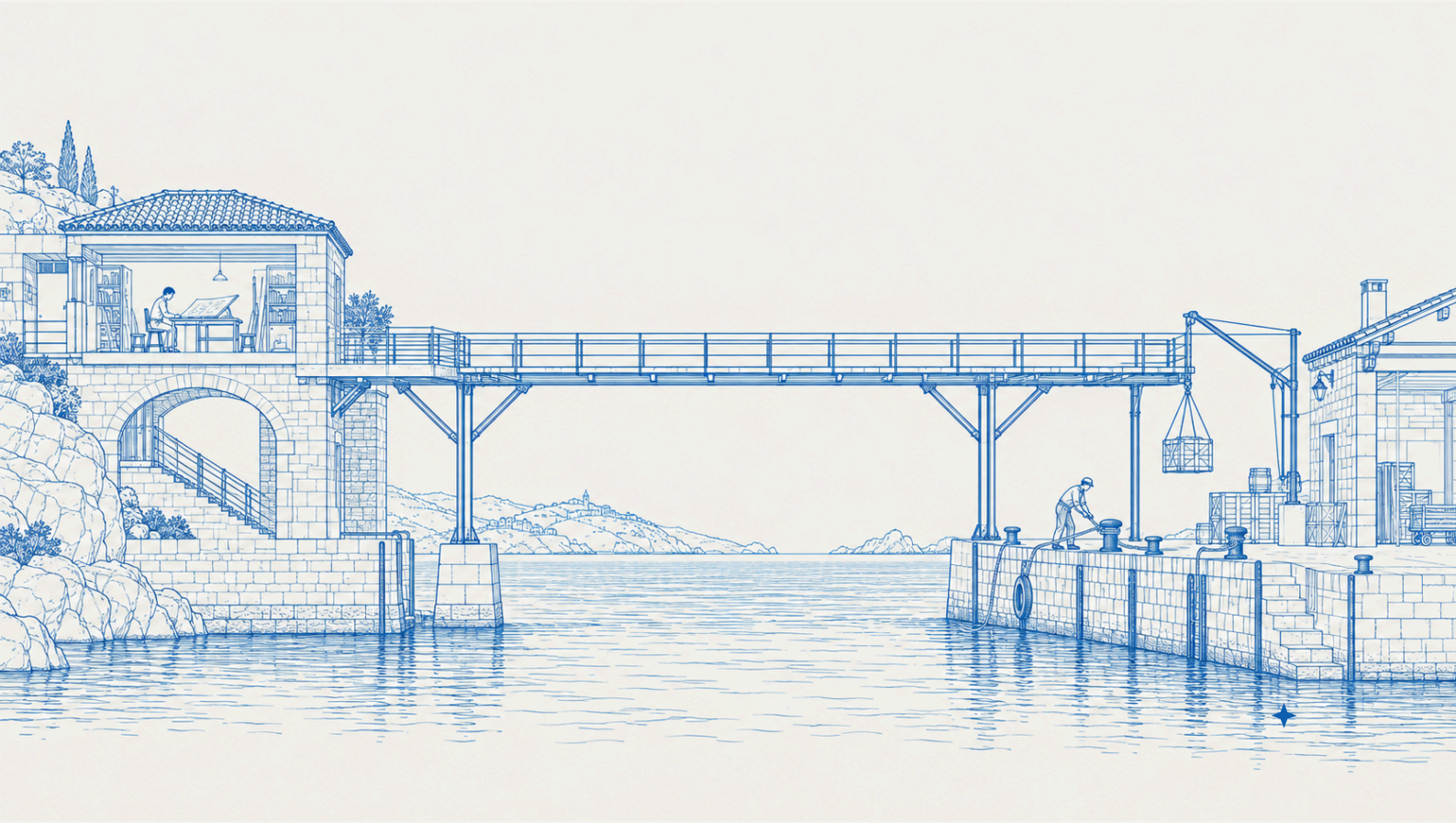
GOVERNED INTELLIGENCE · WHITEPAPER

Agentische KI im Unternehmen gestalten

Eine Architektur für Wissen, Entscheidungen und verlässliche Ausführung

Timotheos Samartzidis

2. November 2026 · Fassung 4.2



Vom überzeugenden Vorschlag zur verlässlichen Handlung

Ein KI-Agent kann einen Bericht lesen, Zusammenhänge erkennen und einen nächsten Schritt vorschlagen. Sobald dieser Vorschlag im Unternehmen etwas bewirken soll, werden einige sehr vertraute Fragen relevant: Welcher Datenstand gilt? Welche Regel ist anzuwenden? Wer trägt die Verantwortung? Und wer darf die Änderung tatsächlich auslösen?

Der Agent mag diese Fragen in tadellosem Deutsch beantworten. Das führende System zeigt sich von seiner Eloquenz gewöhnlich unbeeindruckt. Es braucht gültige Eingaben, passende Berechtigungen und eine eindeutig bestimmte Aktion. Genau an diesem Übergang wird Enterprise Architecture, also die Gestaltung der Zusammenhänge zwischen Geschäft, Daten und IT, für agentische KI konkret. Ein KI-Agent verbindet hier ein Sprachmodell mit Werkzeugen und einer Software, die mehrere Arbeitsschritte ausführen kann. Das Sprachmodell, häufig LLM für Large Language Model genannt, interpretiert den Auftrag. Welche Handlung daraus werden darf, muss die umgebende Architektur festlegen.

Meine Empfehlung ist, Unternehmenswissen, Entscheidungen und Ausführung als ein gemeinsames System zu gestalten. Die Wissensbasis beschreibt die Dinge, über die das Unternehmen spricht, und verbindet sie mit ihren Quellen. Die Entscheidungsarchitektur ordnet Aufgaben den passenden Mechanismen zu: Code berechnet definierte Werte, Regeln prüfen festgelegte Voraussetzungen, und LLMs interpretieren Sprache und entwickeln Vorschläge. Die Ausführungsarchitektur sorgt dafür, dass eine erlaubte Handlung unter den richtigen Bedingungen wirksam wird.

Die agentische Laufzeit ist die Software, die diese Beiträge während der Bearbeitung zusammenbringt. Sie hält den Auftrag über mehrere Schritte hinweg, beschafft Kontext, beauftragt Komponenten und verarbeitet deren Ergebnisse. Dadurch lässt sich nachvollziehen, wie aus einer Information eine Entscheidung und aus einer Entscheidung eine bestätigte Handlung entsteht.

Wenn Sie Unternehmensarchitektur, Daten, Plattformen oder Fachanwendungen verantworten, soll Ihnen dieser Entwurf helfen, einen eigenen Anwendungsfall zu strukturieren. Welche Quellen braucht er? Welche Entscheidungen gehören in Code, Regeln oder menschliche Prüfung? Und welche Zusagen muss die technische Ausführung einlösen? Am Ende steht ein abgegrenzter Architekturauftrag mit benannten Ergebnissen. Einleitung und Wissensmodell erschließen die fachliche Frage; die anschließend markierten technischen Vertiefungen erklären Laufzeit, Aktionsvertrag und Wiederaufnahme.

Dieses Whitepaper entwickelt diesen Zusammenhang vom fachlichen Wissensmodell bis zu einem groben technischen Laufzeitentwurf. Die dargestellten Komponenten und Verträge sind ein Architekturvorschlag, keine bereits implementierte oder getestete Plattform. Der Ausgangspunkt liegt bei einer Voraussetzung, die leicht selbstverständlich wirkt: Alle Beteiligten müssen wissen, wovon sie sprechen.

Unternehmenswissen und seine Quellen verbinden

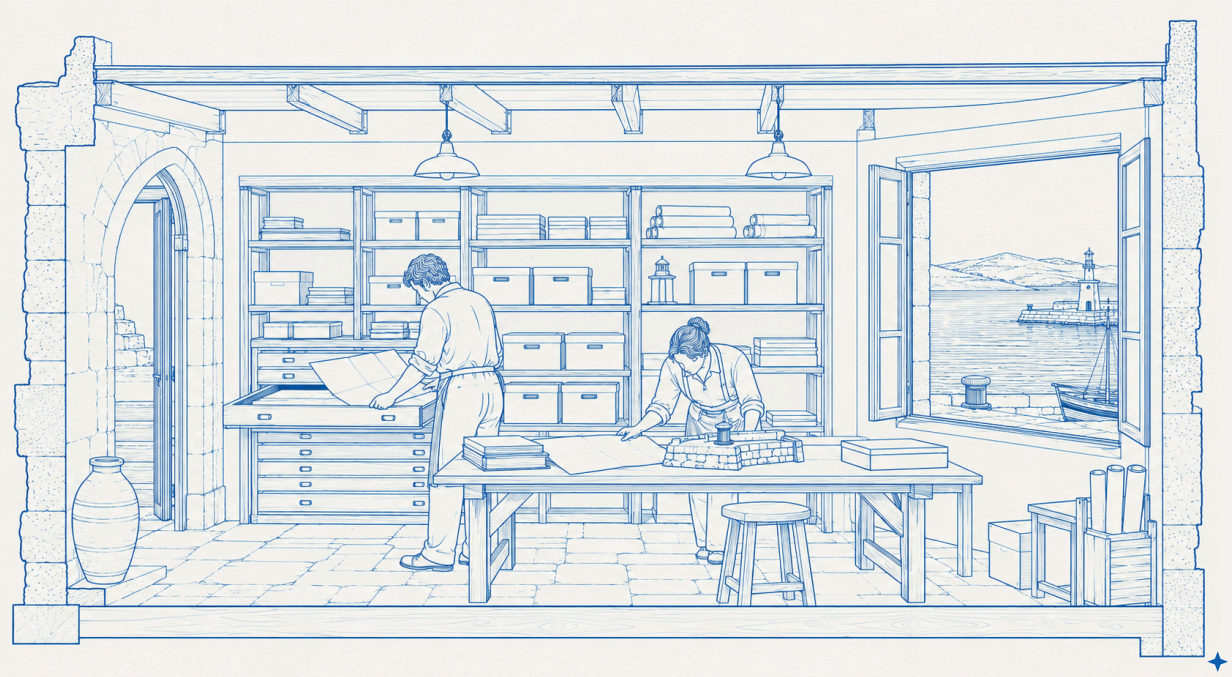
Ein Unternehmen verteilt sein Wissen auf Fachtexte, Datenbanken, Anwendungen und die Menschen, die deren Eigenheiten kennen. Ein Agent braucht einen Zugang zu diesen Zusammenhängen. Dazu muss er erkennen können, welche Begriffe dasselbe meinen, welche Daten zusammengehören und welche Quelle für die jeweilige Frage maßgeblich ist. Die folgenden Bausteine tragen unterschiedliche Teile dazu bei; keiner ersetzt die anderen.

Baustein	Beitrag zum gemeinsamen Wissensmodell
Enterprise Architecture	Fähigkeiten, Prozesse, Anwendungen, Daten, Verantwortungen und Abhängigkeiten zusammenführen.
Wiki und Fachtexte	Erläuterungen, Begründungen, Ausnahmen und fachlichen Kontext erhalten.
Glossar und Taxonomie	Begriffe definieren und Kategorien konsistent ordnen.
Fachliche Ontologie (Business Ontology)	Geschäftsobjekte, Eigenschaften und fachliche Beziehungen maschinenlesbar beschreiben.
Wissensgraph (Knowledge Graph)	Konkrete Objekte und Beziehungen mit ihren Quellen verknüpfen.
Führende Systeme	Verbindliche Datensätze und Transaktionen im jeweiligen Verantwortungsbereich halten.

Ein Glossar erklärt Begriffe; eine Taxonomie ordnet sie in Kategorien. Eine **Ontologie** beschreibt Geschäftsobjekte und ihre fachlichen Beziehungen, etwa wie Vertrag, Risiko und Ereignis zusammenhängen. Ein **Knowledge Graph**, auf Deutsch Wissensgraph, hält konkrete Objekte und bekannte Verbindungen fest. So kann ein Vertragsobjekt auf seine Dokumente, den zuständigen Bereich und den verbindlichen Datensatz verweisen. Das Begriffsmodell und die konkreten Daten bleiben dabei unterscheidbar.

Stabile Identifikatoren, also über Systeme hinweg zuordenbare Kennungen, verbinden diese Sichten. Ein **führendes System** hält den maßgeblichen Datensatz oder führt die verbindliche Transaktion in seinem Verantwortungsbereich aus. Für den einzelnen Auftrag muss feststehen, welche Quelle für welche Aussage führt. Der Zugriff auf Dokumente und Daten bleibt auf den zulässigen Kontext begrenzt; ihre Aufnahme in einen Graphen erweitert keine Berechtigung.

Damit ist Bedeutung beschrieben. Ob eine Aussage heute gilt, muss weiterhin an ihren Quellen geprüft werden. Für entscheidungsrelevante Informationen gehören deshalb Verantwortlicher, fachlicher Gültigkeitszeitraum und Abrufzeit zum Kontext. Beobachtete Fakten, Schätzungen, Hypothesen und Widersprüche bleiben unterscheidbar. Eine Vermutung des Agenten erhält erst durch einen verantworteten Prüfprozess einen anderen Status.



Quellen müssen zugänglich, zuordenbar und verantwortet sein. Fiktives Archiv · generiert

Für die technische Umsetzung stehen unterschiedliche Standards zur Verfügung. SKOS unterstützt Begriffssysteme, OWL formale Ontologien. SHACL beschreibt prüfbare Bedingungen für Daten im RDF-Graphmodell, etwa verlangte Eigenschaften und zulässige Werte. Eine bestandene Strukturprüfung belegt noch keine fachliche Wahrheit. Welche dieser Mittel Sie benötigen, richtet sich nach dem Anwendungsfall und den vorhandenen Systemen.

PRAXISANLEITUNG · ARCHITEKTURAUFTRAG

Wissensausschnitt für eine Entscheidung erfassen

Wählen Sie eine konkrete Entscheidung und erfassen Sie nur deren benötigte Begriffe, Objekte und Quellen. Halten Sie je Information fest: fachlicher Eigentümer, führende Quelle, Kennung, Gültigkeit und erlaubter Zugriff. Markieren Sie ungeklärte Widersprüche. Das erwartete Ergebnis ist ein kleines verantwortetes Wissensmodell mit Quellenzuordnung und offenen Fragen. Ein unternehmensweites Vollmodell ist dafür keine Voraussetzung.

Damit ist festgelegt, worauf sich die Bearbeitung stützen darf. Die nächste Architekturaufgabe besteht darin, dieses Wissen für die jeweilige Entscheidung angemessen zu verwenden.

Welche Aufgaben Code, Regeln und LLMs übernehmen

Wer eine Frist berechnet, eine Regel anwendet oder einen widersprüchlichen Text auslegt, bearbeitet unterschiedliche Aufgaben. Eine tragfähige Entscheidungsarchitektur macht diese Unterschiede ausdrücklich. So kann jede Komponente dort eingesetzt werden, wo ihr Ergebnis sinnvoll prüfbar ist.

Mechanismus	Geeignete Aufgabe	Was das Ergebnis trägt
Deterministischer Code	Beträge, Fristen und definierte Funktionen berechnen.	Reproduzierbarkeit bei gleichen Eingaben, gleichem Zustand und gleicher Implementierung.
Regeln / DMN (Decision Model and Notation)	Explizite Voraussetzungen und Entscheidungstabellen auswerten.	Anwendung einer dokumentierten Regel auf einen bestimmten Eingabestand.
LLM-Interpretation	Sprache auswerten, Widersprüche erkennen und nächste Schritte vorschlagen.	Ein prüfbarer Vorschlag innerhalb eines begrenzten Auftrags.

Decision Model and Notation, kurz DMN, ist ein Standard für die Modellierung von Geschäftsentscheidungen, unter anderem mit Entscheidungstabellen. Eine Regel kann etwa eine ausdrücklich festgelegte Voraussetzung prüfen. Ein Sprachmodell kann zuvor aus einem Dokument einen prüfbedürftigen Sachverhalt erschließen. Für die nachfolgende Berechnung eignet sich definierter Code. Die Architektur legt fest, wie solche Beiträge ineinandergreifen und an welcher Stelle ihr Ergebnis bestätigt werden muss.

Bei einem LLM-Ergebnis entsteht fachliche Verbindlichkeit durch die vorgesehene Prüfung und Autorisierung. Wie überzeugend es formuliert ist, beantwortet diese Frage noch nicht. Auch eine korrekt ausgewertete Regel liefert nur dann die gewünschte Entscheidung, wenn Regelstand und Eingaben passen.

Zur fachlichen Entscheidung kommt die Frage, ob die konkrete Handlung zulässig ist. Eine **Policy** ist hier eine technisch auswertbare Vorgabe zu Befugnissen und Bedingungen. Eine **Policy Engine** entscheidet anhand strukturierter Angaben, ob diese Vorgabe erfüllt ist; der ausführende Dienst setzt das Ergebnis durch. Open Policy Agent trennt diese beiden Verantwortlichkeiten entsprechend.

Hinzu kommt die Tragweite. Jeff Bezos unterscheidet gut reversible Type-2- von schwer umkehrbaren Type-1-Entscheidungen. Diese Dimension hilft, die erforderliche Prüfung zu bestimmen. Sie legt für sich genommen noch nicht fest, welches Werkzeug an der Vorbereitung mitwirken darf.

Im folgenden Entwurf richten sich menschliche Freigaben nach fachlicher Verantwortung, Unsicherheit und möglichen Folgen. Eine Freigabe gilt für eine konkrete Handlung unter bestimmten Bedingungen. Zusammen mit dem Wissensmodell ergibt sich daraus die übergreifende Architektur.

Wie Wissensbasis, Orchestrierung und Fachsystem zusammenwirken

Wissen, Entscheidungsmechanismen und ausführende Systeme müssen sich aufeinander beziehen können. Abbildung 1 ordnet ihre logischen Verantwortlichkeiten. Lesen Sie sie von oben nach unten: Ein geschäftlicher Auftrag nutzt die gemeinsame Wissensbasis; die Orchestrierung koordiniert die Bearbeitung und wählt passende Mechanismen; vor der wirksamen Änderung steht die Prüfung der konkreten Aktion. Das Fachsystem bestätigt schließlich ihr Ergebnis. Produkte, Netzwerkpfade und die Verteilung auf einzelne Dienste bleiben offen.

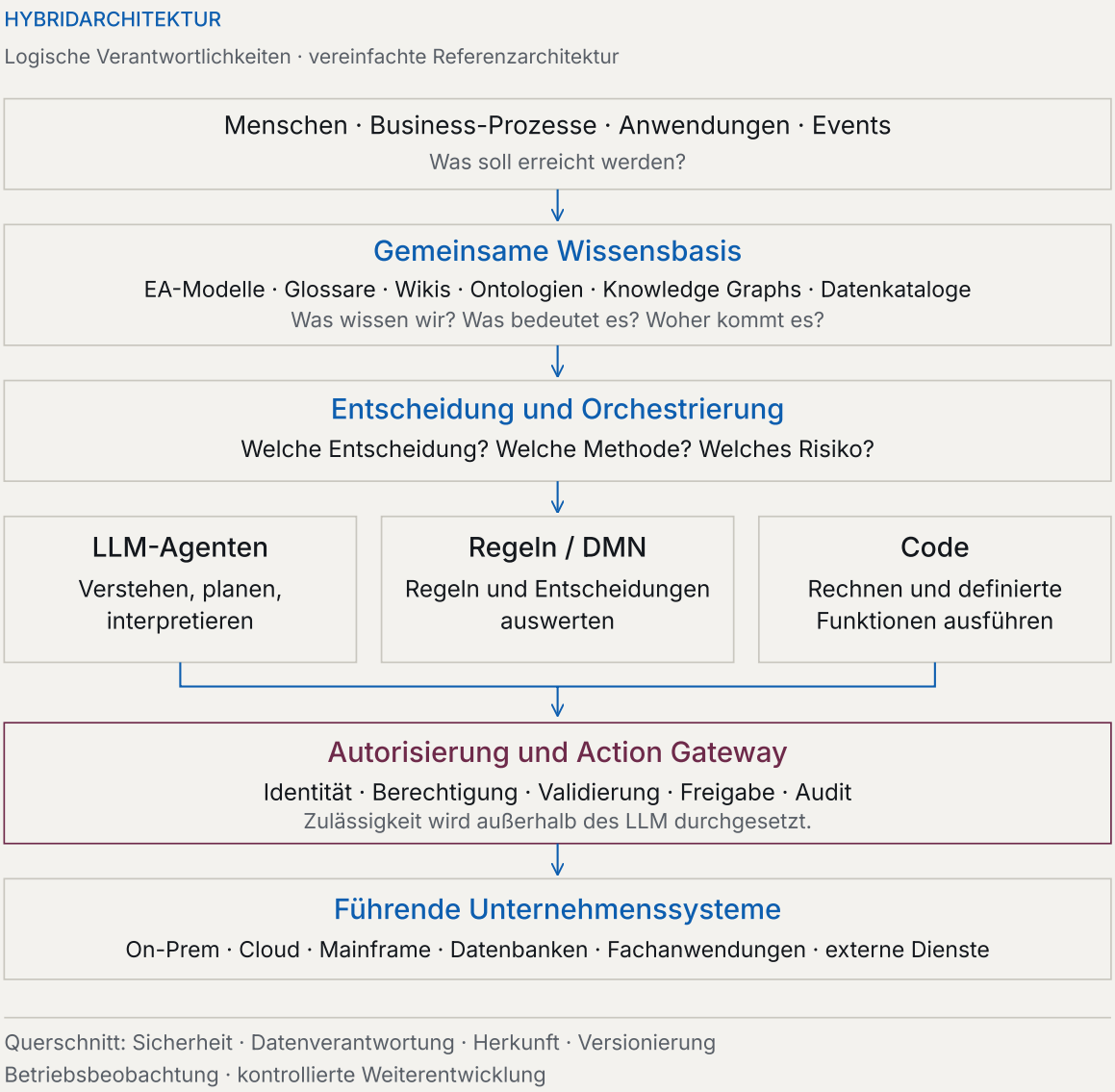
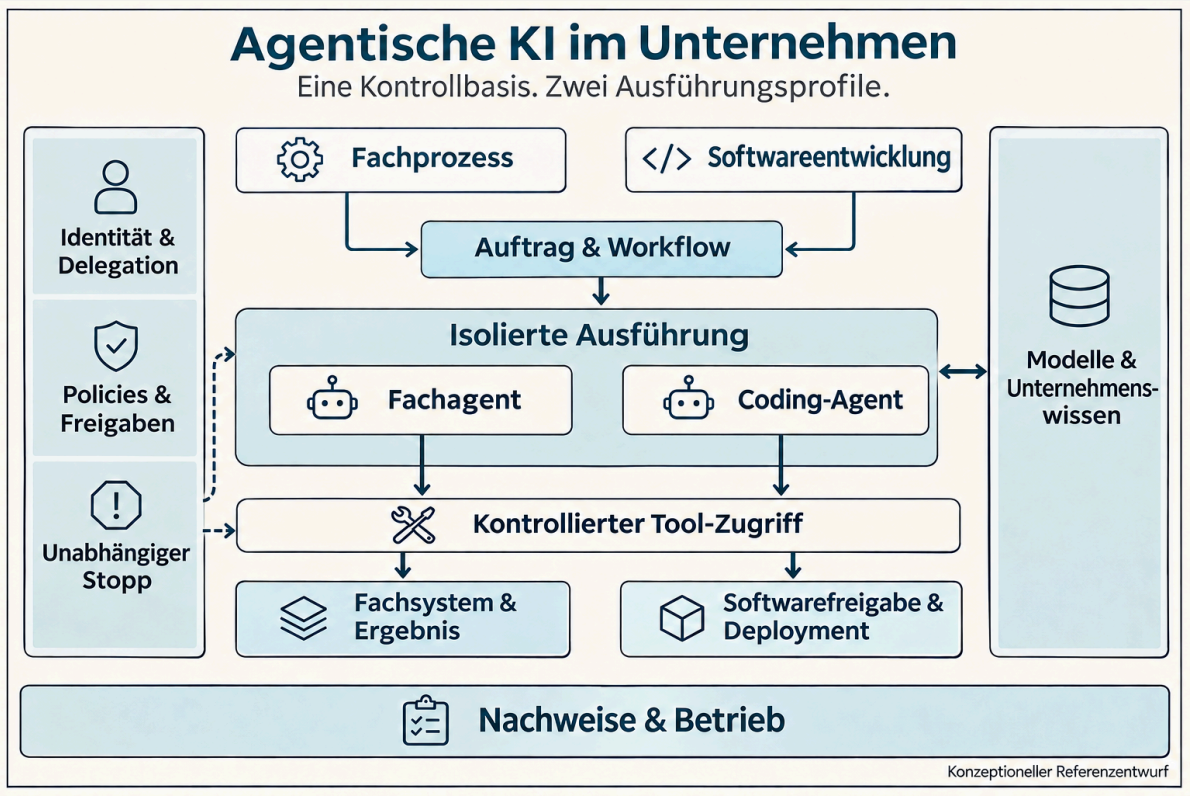


Abbildung 1. Logische Verantwortlichkeiten vom Auftrag bis zum Fachsystem. Pfeile zeigen den konzeptionellen Zusammenhang; Querschnittsaufgaben gelten für alle Ebenen. Eigener Referenzentwurf.

Die Pfeile zeigen diesen logischen Zusammenhang. Sie sind keine Netzverbindungen und beschreiben keine bereits eingerichteten Sicherheitsgrenzen. Die unterste Zeile benennt Aufgaben für alle Ebenen: Sicherheit und verantwortete Datenpflege, die Herkunft von Informationen, Versionsstände, die Beobachtung des Betriebs und die kontrollierte Weiterentwicklung.

Die gemeinsame Bedeutung hält die Ebenen zusammen. Ändert sich eine Fachregel, muss ihre Auswirkung auf den Kontext des Agenten, die Regelauswertung und die betroffenen Aktionen geprüft werden. Enterprise Architecture macht diese Beziehungen nachvollziehbar. Fachliche Eigentümer verantworten die Inhalte; Engineering und Betrieb sorgen für deren technischen Vollzug.



Ergänzender Referenzentwurf: Zwei Ausführungsprofile nutzen gemeinsame Kontrollen. KI-generiertes Diagramm.

Verwandte Bausteine finden sich in der Palantir-Ontologiearchitektur und den AWS-Leitfäden zu agentischer Enterprise-Architektur. Ihre Verbindung in dieser Darstellung ist ein eigener Referenzentwurf. Die Ähnlichkeit einzelner Bausteine belegt keine identische Umsetzung und keine Wirkung des Gesamtentwurfs. Wie die Rollen während eines Auftrags zusammenarbeiten sollen, erschließt die folgende Laufzeitskizze.

Wie die Laufzeit einen Auftrag über mehrere Schritte führt

TECHNISCHE VERTIEFUNG

Komponenten und ihre Aufgaben

Zwischen einem Auftrag und seinem Ergebnis liegen oft mehrere Schritte, Rückfragen und Wartezeiten. Die Laufzeit muss diesen Zusammenhang erhalten. Ein **Run** ist hier die zusammengehörige Bearbeitung eines Auftrags mit eigener Kennung und gespeichertem Zustand. Er kann viele Modell- und Werkzeugaufrufe umfassen und länger bestehen als ein einzelner Aufruf.

Die folgenden englischen Bezeichnungen benennen Rollen im Entwurf. Sie schreiben keine Produkte und auch keinen eigenen Dienst pro Kasten vor. Bereits vorhandene Workflow-, Integrations- oder Anwendungsdienste können mehrere dieser Aufgaben übernehmen, wenn die nötigen Zusagen überprüfbar bleiben.

Rolle im Entwurf	Aufgabe während eines Runs
Run Manager	Den Aufrufer prüfen und Auftrag, delegierte Befugnisse, Kennung und Grenzen zusammenführen.
Ereigniswarteschlange	Neue Arbeit, eingetroffene Ergebnisse und erlaubte Fortsetzungen zur Bearbeitung bereitstellen.
Orchestrator	Den nächsten begrenzten Schritt koordinieren und Zustandsübergänge kontrollieren.
Context Builder	Die für den Auftrag erlaubten Informationen mit Herkunft und Datenstand zusammenstellen.
Run Store	Zustand, offene Aktionen, Freigaben und Nachweise dauerhaft speichern.
Worker	Einen begrenzten Auftrag mit LLM, Regeln oder Code bearbeiten und ein Ergebnis zurückgeben.
Action Gateway	Eine konkret vorgeschlagene Änderung vor ihrer Weitergabe an das Fachsystem prüfen.

Ein **Action Intent** beschreibt die beabsichtigte Änderung mit Ziel, Eingaben und Ausführungskontext. Ein **Tool-Adapter** übersetzt den autorisierten Auftrag in den Aufruf der jeweiligen Programmierschnittstelle, kurz API. Mit diesen Rollen zeigt Abbildung 2, wie ein Run bis zur bestätigten Handlung gelangen soll.

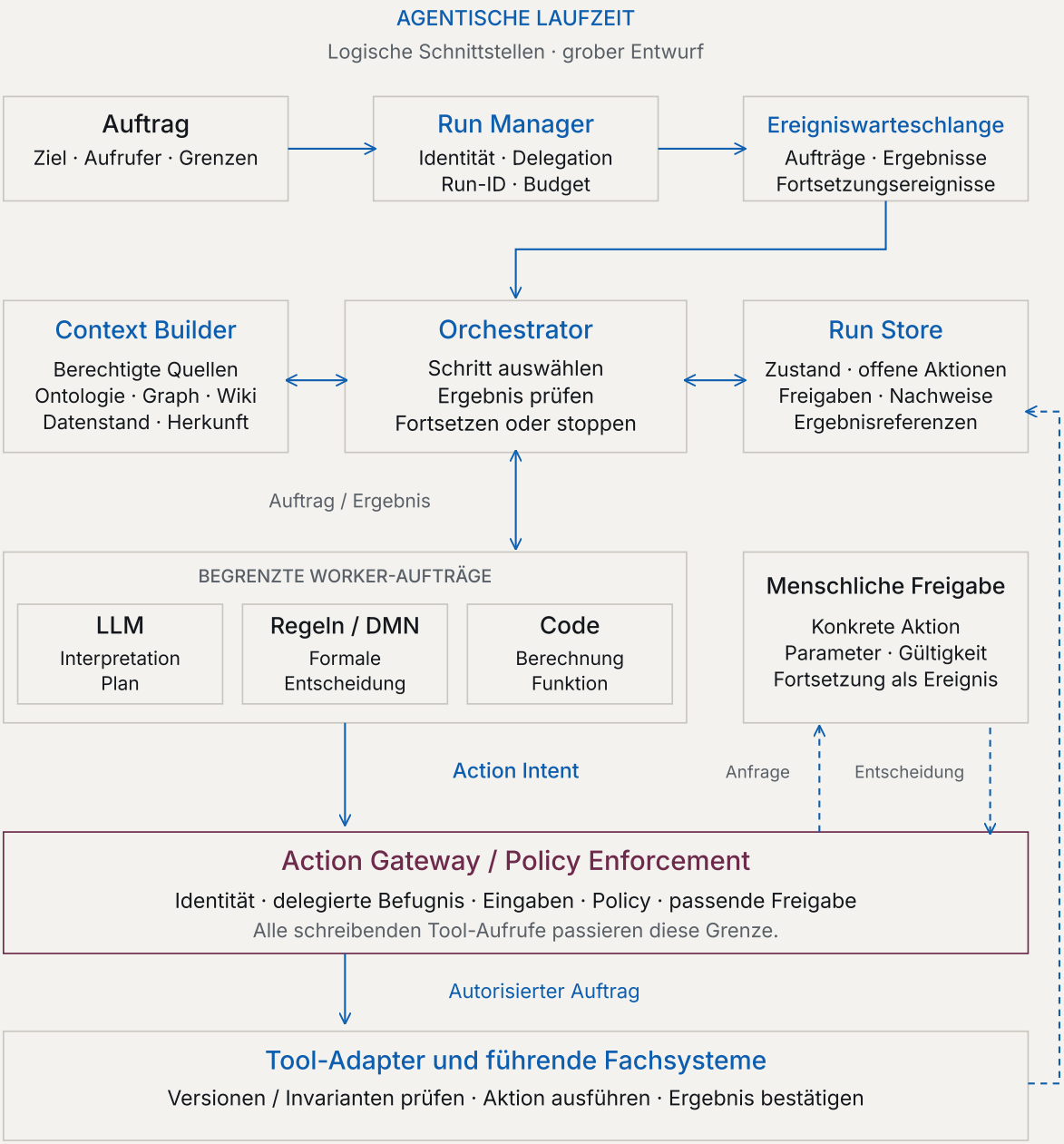


Abbildung 2. Rollen und logische Schnittstellen der vorgeschlagenen Laufzeit. Durchgezogene Pfeile: Arbeitsauftrag oder Rückgabe. Gestrichelt: Freigabe oder Ergebnisnachweis. Keine geprüfte Implementierung.

Durchgezogene Pfeile zeigen Arbeitsaufträge und Rückgaben, gestrichelte Pfeile Freigaben oder Ergebnisnachweise. Kontextbeschaffung und gespeicherter Bearbeitungszustand stehen dem Orchestrator als getrennte Aufgaben zur Verfügung. Der rechte Rückweg hält die Bestätigung aus dem Fachsystem am Vorgang fest. Eine später eintreffende Freigabe kann über ein Fortsetzungsereignis neue Bearbeitung auslösen; sie führt die Aktion nicht selbst aus.

Die hervorgehobene Gateway-Grenze ist eine Anforderung an die Umsetzung: Schreibende Werkzeugaufrufe müssen diese Prüfung durchlaufen. Dafür dürfen die Worker keine gleichwertigen Schreibzugänge erhalten, die sie umgehen. Wie Berechtigungen, Laufzeitisolation und Schnittstellen diese Grenze tatsächlich durchsetzen, bleibt im konkreten technischen Entwurf nachzuweisen. Der Ablauf innerhalb dieser Rollen lässt sich nun Schritt für Schritt verfolgen.

Vom Auftrag zum nächsten bestätigten Schritt

Ein Agent braucht Raum, um einen sinnvollen nächsten Schritt zu finden. Die Laufzeit braucht klare Bedingungen, um diesen Schritt zuzulassen. Der Orchestrator verbindet beides in einem wiederkehrenden Zyklus und kontrolliert dessen Übergänge außerhalb des LLM.

1. **Den Auftrag binden.** Der Run Manager authentifiziert den Aufrufer und bestimmt die delegierten Befugnisse. Ziel, Budget und Abbruchbedingungen erhalten mit der Run-ID einen gemeinsamen Bezug.
2. **Den Zustand übernehmen.** Der Orchestrator nimmt einen neuen oder fortzusetzenden Auftrag aus der Warteschlange. Ein exklusiver Bearbeitungsanspruch und versionsgeprüfte Zustandsänderungen sollen konkurrierende Fortschreibungen desselben Runs abwehren. Die Umsetzung muss auch verhindern, dass ein veralteter Bearbeiter nach Verlust seines Anspruchs weiter gültige Änderungen schreibt.
3. **Den Kontext beschaffen.** Der Context Builder liest die zulässigen Fachobjekte und Quellen. Datenstand, Herkunft und offene Fragen begleiten den Arbeitsauftrag.
4. **Den nächsten Schritt bestimmen.** LLM, Regelkomponente oder Code liefern ein strukturiertes Ergebnis. Die Laufzeit prüft Schema, fachliche Voraussetzungen und zulässigen Aufgabenumfang.
5. **Ausführen oder warten.** Lesende Werkzeuge arbeiten im Zugriffskontext des Runs. Für eine schreibende Aktion entsteht ein Action Intent. Fehlen Informationen oder eine Freigabe, speichert die Laufzeit den Wartezustand dauerhaft.
6. **Die Handlung autorisieren.** Das Gateway prüft den konkreten Intent gegen aktuelle Befugnisse, Regeln und die gegebenenfalls nötige Freigabe. Geänderte Parameter oder relevante Datenstände führen zu einer erneuten Prüfung.
7. **Das Ergebnis bestätigen.** Der Adapter meldet Erfolg oder Fehler. Bleibt der Ausgang unklar, beginnt der Abgleich mit dem Zielsystem. Erst ein bestätigtes Ergebnis verändert den bekannten Ausführungsstand.
8. **Fortsetzen oder beenden.** Der Orchestrator prüft Ziel und Grenzen. Ein weiterer Schritt braucht verbleibendes Budget und einen zulässigen Zustandsübergang.



Beispiel Zahlung: Fachentscheidung, gegebenenfalls Freigabe und verbindliche Ausführung bleiben getrennt. KI-generiertes Diagramm.

Das LLM entwickelt innerhalb dieses Rahmens seine Vorschläge. Schrittzahl, Fristen, Kosten und Abbruch kontrolliert die Laufzeit. Sobald ein Vorschlag eine verbindliche Änderung auslösen soll, muss er dafür eindeutig beschrieben sein. Diese Aufgabe übernimmt der Aktionsvertrag.

Der Aktionsvertrag legt Ziel, Eingaben und Freigabe fest

TECHNISCHE VERTIEFUNG

Felder des Aktionsvertrags verstehen

„Führe den nächsten sinnvollen Schritt aus“ reicht als Auftrag an ein Fachsystem kaum aus. Es muss feststehen, welche Aktion welches Objekt mit welchen Eingaben verändern soll. Ein Action Intent hält diese Angaben zusammen mit dem Ausführungskontext fest. Das folgende JSON (JavaScript Object Notation, ein strukturiertes Datenformat) skizziert diesen Vertrag. Es enthält schematische Werte und Platzhalter. Es ist weder ein vollständiges Schnittstellenschema noch eine ausführbare Implementierung.

```
{
  "runId": "run-17",
  "subjectRef": "actor-12",
  "delegationRef": "delegation-4",
  "actionId": "action-7",
  "targetRef": "record-42",
  "expectedVersion": "v19",
  "inputRef": "input-3",
  "inputHash": "sha256:…",
  "policyRevision": "policy-6",
  "approvalRef": null,
  "idempotencyKey": "record-42:action-7",
  "status": "PROPOSED"
}
```

Im Beispiel gehört die vorgeschlagene Aktion `action-7` zum Run `run-17` und betrifft `record-42`. Die Eingaben liegen hinter `input-3`; sie sind hier nicht ausgeschrieben. `PROPOSED` bedeutet, dass die Handlung vorgeschlagen wurde. Eine Ausführung ist damit nicht belegt. Die Feldnamen sollen die folgenden Fragen zusammenhalten:

Felder	Bedeutung und verantwortliche Prüfung
`runId`	Zu welchem Vorgang gehört die Aktion? Die Laufzeit ordnet sie dem gespeicherten Run zu.
`subjectRef`, `delegationRef`	Wer handelt in wessen Auftrag? Vertrauenswürdige Identitäts- und Laufzeitdienste liefern die Referenzen; das Gateway prüft die Befugnis.
`actionId`, `targetRef`, `inputRef`	Welche konkrete Aktion betrifft welches Ziel mit welchen Eingaben? Adapter und Gateway müssen Aktionstyp und Inhalte eindeutig auflösen und validieren können.
`expectedVersion`, `policyRevision`	Welcher Objekt- und Regelstand liegt zugrunde? Das Gateway prüft die anwendbare Policy; das Zielsystem prüft seine maßgebliche Objektversion.
`inputHash`	Auf welchen Eingabestand bezieht sich die Prüfung? Ein Hash ist ein aus den Daten berechneter Prüfwert. Berechnung und Bindung an die tatsächlich ausgeführten Eingaben sind festzulegen.
`approvalRef`	Wo ist die gegebenenfalls nötige Freigabe belegt? `null` bedeutet hier, dass keine Referenz eingetragen ist. Ob ohne Freigabe ausgeführt werden darf, entscheidet die geltende Vorgabe.
`idempotencyKey`	Welche Wiederholungen gehören zur selben logischen Aktion? Die Laufzeit vergibt den Schlüssel; der ausführende Dienst muss ihn auswerten. Das Zielsystem muss doppelte Geschäftswirkungen zuverlässig abwehren können.

Felder	Bedeutung und verantwortliche Prüfung
`status`	Welcher Aktionsstand ist bestätigt? Zustandsänderungen erfolgen durch die Laufzeit anhand geprüfter Ereignisse.

`actionId` setzt im Beispiel eine eindeutige Auflösung auf eine registrierte Aktion voraus. Ein reales Schnittstellenschema muss zusätzlich festlegen, wie Aktionstyp, Parameterstruktur und erlaubte Zielobjekte bestimmt werden. Der verkürzte Wert `sha256:…` ist kein tatsächlicher Hash. Auch der Idempotenzschlüssel ist ein Beispiel: Für unterschiedliche Geschäftshandlungen darf nicht versehentlich derselbe Schlüssel entstehen.

Der Agent schlägt die Handlung vor und liefert seine fachlichen Argumente. Vertrauenswürdige Laufzeitkomponenten ergänzen den verifizierten Aufruferkontext, Befugnisreferenzen und Nachweise. Eine Berechtigung, die das Modell sich selbst zuschreibt, hat dabei ungefähr den Beweiswert einer selbst ausgestellten Vollmacht.

Eine Freigabe bindet Aktion, Eingaben, Ziel und relevante Versionen. Ändert sich einer dieser Bestandteile, reicht die frühere Freigabe nicht mehr aus. Unmittelbar vor der verbindlichen Änderung prüft das Zielsystem seine maßgeblichen Bedingungen und Versionen innerhalb der Transaktion. Zwischenzeitliche Änderungen lassen sich durch eine vorgelagerte Gateway-Prüfung allein nicht ausschließen.

Das Model Context Protocol, kurz MCP, beschreibt unter anderem, wie Werkzeuge mit Eingabeschemata und gegebenenfalls Ausgabeschemata angeboten werden. Solche Beschreibungen helfen beim passenden Aufruf. Für den hier entworfenen Aktionsvertrag müssen darüber hinaus Identität, Freigabebindung und bestätigtes Ergebnis durch die beteiligten Dienste durchgesetzt werden.

PRAXISANLEITUNG · TECHNISCHER ARBEITSAUFTRAG

Aktionsvertrag mit dem Zielsystem abstimmen

Nehmen Sie eine einzelne schreibende Aktion und klären Sie mit dem Anwendungsteam jedes Feld des Beispiels: Wer setzt es, wo wird es gespeichert und wer prüft es? Ergänzen Sie Aktionstyp, Schema, Fehlerantworten und die Bindung einer Freigabe. Dokumentieren Sie, wie das Zielsystem Versionen und wiederholte Aufrufe behandelt. Das erwartete Ergebnis ist ein abgestimmter Schnittstellenvertrag mit offenen Implementierungsfragen. Das JSON allein erfüllt diesen Auftrag noch nicht.

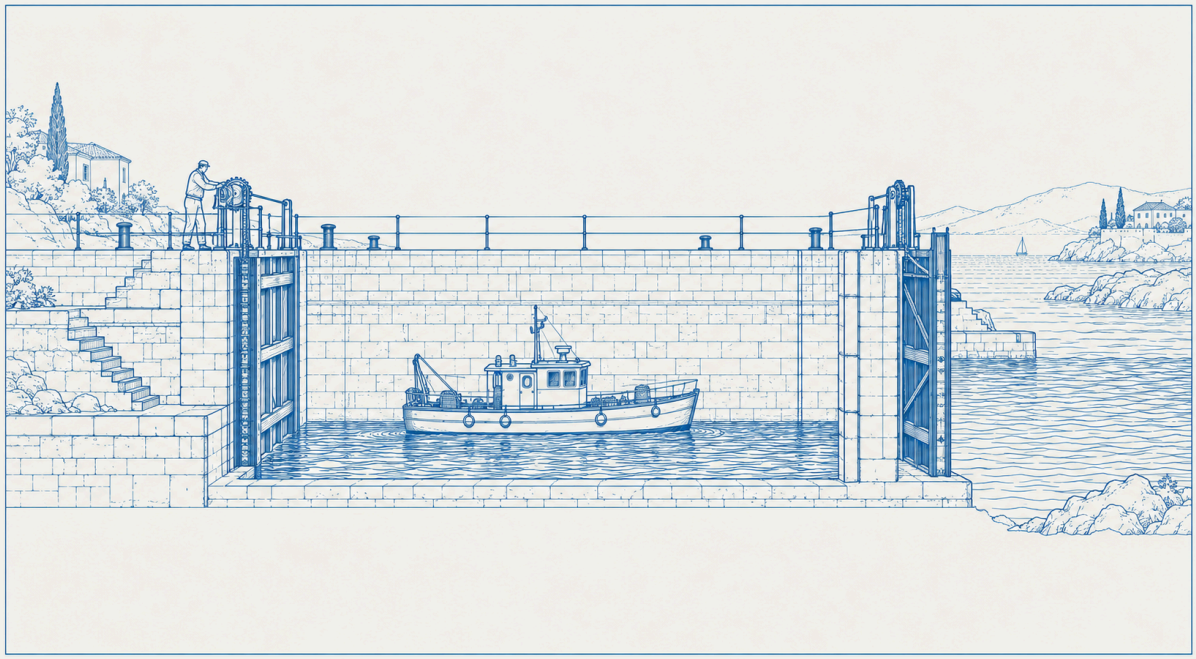
Ihre Durchsetzung bleibt damit Aufgabe der Architektur. Dass zwischen Vorschlag und Ausführung Zeit vergehen kann, führt zur nächsten Anforderung: Der Vorgang muss eine Unterbrechung überstehen.

Aufträge speichern, offene Ausgänge klären und fortsetzen

TECHNISCHE VERTIEFUNG

Gespeicherter Zustand und Wiederaufnahme

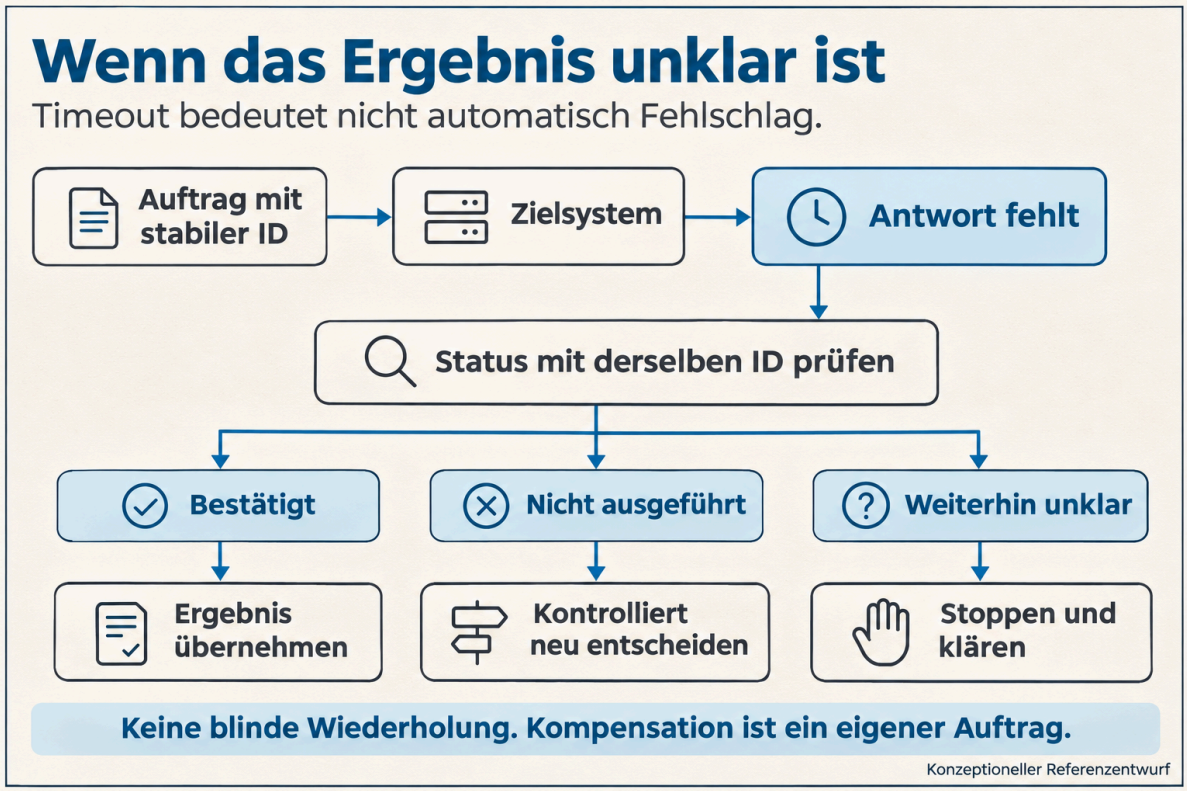
Eine Freigabe kann eintreffen, wenn der ursprüngliche Modellaufruf längst beendet ist. Ein Dienst kann eine Änderung vorgenommen haben, obwohl seine Antwort ausbleibt. Für beide Situationen braucht die Laufzeit einen gespeicherten Zustand, aus dem eine zulässige Fortsetzung hervorgeht.



Auch nach einer Wartezeit müssen die Bedingungen für den nächsten Schritt stimmen. Fiktive Schleuse · generiert

Run-Zustand	Bedeutung und mögliche Fortsetzung
QUEUED / RUNNING	Der Auftrag wartet auf Bearbeitung oder führt einen begrenzten Schritt aus.
WAITING_INPUT	Eine benannte Information fehlt; ein passendes Ereignis kann den Run fortsetzen.
WAITING_APPROVAL	Eine konkrete Handlung wartet auf eine befugte Entscheidung.
RECONCILING	Der Ausgang einer Aktion wird geklärt; abhängige Aktionen bleiben gesperrt.
COMPLETED	Das festgelegte Ziel ist mit bestätigtem Ergebnis erreicht.
STOPPED / FAILED	Abbruchbedingung oder Fehler verhindert die automatische Fortsetzung; der Grund bleibt erhalten.

Diese Zustände sind ein Designvorschlag für den Run. Einzelne Aktionen haben einen eigenen Verlauf von Vorschlag über Prüfung und Versand bis zur Bestätigung. **UNKNOWN** bezeichnet einen nach Versand ungeklärten Ausgang. Ein Timeout bestätigt zunächst, dass eine Antwort fehlt. Über den Erfolg der Aktion äußert er sich mit bemerkenswerter Zurückhaltung.



Bei unklarem Ausgang zuerst abgleichen; eine Kompensation ist ein eigener Auftrag. KI-generiertes Diagramm.

Bei einer späteren Freigabe lädt die Laufzeit die gespeicherte Aktion und prüft Gültigkeit, unveränderte Parameter und aktuelle Entscheidungslage erneut. Ein gespeicherter Verlauf macht Wiederaufnahme möglich; frühere Berechtigungen müssen weiterhin gültig sein.

Bei einer Wiederholung soll dieselbe logische Aktion keine zusätzliche Geschäftswirkung auslösen. Diese Eigenschaft heißt **Idempotenz**. Dazu verwendet der Entwurf denselben Idempotenzschlüssel für dieselbe Aktion; ein neuer fachlicher Auftrag erhält eine eigene Zuordnung. Der ausführende Dienst muss doppelte Aufrufe erkennen; das Zielsystem muss zusätzliche Geschäftswirkungen bei Wiederholung zuverlässig abwehren können. Der Schlüssel allein schafft diese Eigenschaft nicht. Diesen Zusammenhang erläutert auch die Temporal-Dokumentation zu Activities. Ein persistierter Workflow allein garantiert keine einmalige Wirkung in beliebigen Fremdsystemen.

Lässt sich ein Ausgang weder klären noch sicher wiederholen, bleibt die Aktion zur Prüfung stehen. Auch ein Stopp des Runs macht eine bereits bestätigte Geschäftswirkung nicht rückgängig. Eine **Kompensation**, etwa eine fachlich zulässige Gegenbuchung, ist eine eigene, erneut zu autorisierende Handlung. Solche Grenzen gehören zum Laufzeitentwurf und müssen sich im Betrieb überprüfen lassen.

Die Ausführung mit Kontrollen und Nachweisen prüfen

Die Qualität der Architektur zeigt sich an ihren Übergängen: vom Auftrag zum Kontext, vom Vorschlag zur Freigabe und von der Ausführung zum bestätigten Ergebnis. Für jeden Übergang muss erkennbar sein, welche Voraussetzung gilt und was geschieht, wenn sie fehlt.

Kontrollpunkt	Verhalten bei fehlender Voraussetzung
Identität und delegierter Auftrag	Einen Run ohne verifizierten Aufrufer und begrenzte Befugnisse ablehnen.
Kontext und Datenstand	Fehlende, veraltete oder widersprüchliche Angaben kennzeichnen; bei Bedarf Rückfrage oder Fachprüfung.
Ergebnis- und Werkzeugschema	Unpassende Ausgaben zurückweisen; formal gültige Ergebnisse bleiben fachlich prüfpflichtig.
Policy und Freigabe	Unzulässige Aktionen blockieren; eine fehlende Freigabe als Wartezustand führen.
Versionen im Zielsystem	Änderungen bei verletzten Bedingungen oder Versionskonflikten zurückweisen.
Budget und Abbruch	Weitere Schritte bei ausgeschöpftem Budget, Fristüberschreitung oder Stoppentscheidung verhindern.
Unklarer Aktionsausgang	Status abgleichen und abhängige Handlungen bis zur Klärung sperren.

Der Nachweis verbindet den Run mit Aufgabe, Quellen und Datenstand, Modell- und Vorlagenversion, Regeln, Policies, Freigaben und Tool-Aufrufen. Er hält das bestätigte Ergebnis samt überprüfbarer Begründung fest. Dafür braucht es relevante Belege; interne Gedankengänge des Modells sind kein notwendiger Nachweis. Zugriffe, Aufbewahrung und schutzbedürftige Inhalte dieser Aufzeichnungen sind selbst Teil des technischen Entwurfs. Vollständige Dokumentinhalte oder Geheimnisse gehören nicht allein deshalb in ein Protokoll, weil sie im Kontext verfügbar waren.

Damit bleiben auch die Verantwortlichkeiten sichtbar. Fachliche Eigentümer pflegen Begriffe und Regeln. Befugte Entscheider tragen Freigaben. Engineering und Betrieb implementieren die Kontrollen und sorgen für ihren verlässlichen Vollzug. Enterprise Architecture hält die Beziehungen zwischen diesen Verantwortlichkeiten über Änderungen hinweg nachvollziehbar.

Diese Kontrollpunkte beschreiben Anforderungen. Ob eine Implementierung sie erfüllt, muss mit konkreten Nachweisen geprüft werden. Der Entwurf lässt Produktwahl, Schnittstellen, Speicher, Isolation und Leistungsdimensionierung offen. Das ist der Übergang von der allgemeinen Architektur zum eigenen Umsetzungsauftrag.

Aus dem Entwurf einen prüfbaren Auftrag machen

Für einen Einstieg brauchen Sie einen Vorgang, dessen Ziel und Grenzen Ihr Team benennen kann. Wählen Sie eine begrenzte Entscheidung mit überschaubaren Quellen und einer klar beschreibbaren möglichen Handlung. Arbeiten Sie anschließend die Wissens-, Entscheidungs- und Ausführungssicht an genau diesem Ausschnitt aus. Bestehende Dienste und Zuständigkeiten gehören von Anfang an dazu; ein neues Kästchen im Diagramm ist noch kein Grund für einen neuen Dienst.

Das Ergebnis dieser Arbeit sollte für Fachbereich, Architektur, Engineering und Betrieb gemeinsam lesbar sein. Die folgende Zuordnung macht sichtbar, welche Unterlagen vor einer Implementierung fehlen können. Die Namen sind Vorschläge für die Ablage, keine Ausgaben eines bereits ausgeführten Werkzeugs.

Erwartetes Artefakt	Inhalt und verantwortete Entscheidung
Anwendungsfall und Wissensmodell	Ziel, Grenzen, Begriffe, führende Quellen, Zugriffe und fachliche Eigentümer. Der Fachbereich bestätigt die Bedeutung und die verwendbaren Datenstände.
Entscheidungskarte	Zuordnung der Teilschritte zu Code, Regeln, LLM und menschlicher Prüfung samt Begründung und benötigten Eingaben.
Aktions- und Zustandsvertrag	Vertrauenswürdige Herkunft der Felder, Freigabebindung, Versionsprüfung, Wiederholung, Wartezustände und Behandlung unklarer Ergebnisse. Engineering und Anwendungsteam klären die Schnittstellen.
Kontroll- und Betriebsplan	Nachweise für die Kontrollpunkte, Budgets, Abbruch, Protokollzugriff und verantwortliche Reaktion auf Fehler oder einen ungeklärten Aktionsausgang.
Offene Architekturentscheidungen	Produktauswahl, Bereitstellung, Isolation, Datenhaltung und Dimensionierung mit jeweils benannter Verantwortung.

PRAXISANLEITUNG · VORGESCHLAGENE ABNAHMEFÄLLE

Freigabe und Wiederaufnahme in Störfällen prüfen

Prüfen Sie in einer entbehrlichen Testumgebung, was bei fehlender Delegation, geänderten Parametern nach Freigabe, einem Versionskonflikt oder einem doppelten Aufruf geschieht. Unterbrechen Sie außerdem einen Vorgang nach dem Versand, aber vor der Antwort: Die Wiederaufnahme muss den Ausgang klären und darf keine abhängige Handlung ungeprüft fortsetzen. Halten Sie Erwartung, beobachtetes Verhalten und Belege je Fall fest. Diese vorgeschlagenen Abnahmefälle wurden für den vorliegenden Referenzentwurf nicht ausgeführt.

Damit erhält das Team eine Grundlage für die technische Ausarbeitung und eine spätere Abnahme. Für einen Einsatz müssen außerdem Betrieb, Schutzbedarf und Leistung am konkreten System bewertet werden. Eine bestandene Prüfung eines einzelnen Übergangs belegt noch nicht die Verlässlichkeit des gesamten Prozesses. Sie macht aber eine bestimmte Zusage überprüfbar und zeigt, welche Fragen offen bleiben.

Fazit

Ein überzeugender Vorschlag wird erst dann zu einer verlässlichen Handlung, wenn Bedeutung, Entscheidungsweg und Ausführung zusammenpassen. Der Agent braucht verantwortete Quellen; jeder Teilschritt braucht einen geeigneten Mechanismus; die Änderung im Fachsystem braucht eine konkrete Befugnis und ein bestätigtes Ergebnis. Enterprise Architecture verbindet diese Zusagen über Zuständigkeiten und Änderungen hinweg.

Für Ihren eigenen Entwurf lohnt es sich deshalb, einen begrenzten Auftrag vollständig durchzugehen. Klären Sie, welche Information woher stammt, wer welche Entscheidung trägt und unter welchen Bedingungen eine Handlung ausgeführt, wiederholt oder angehalten werden darf. Die dabei entstehenden Wissensmodelle, Verträge und Prüffälle geben der Umsetzung einen belastbaren Arbeitsauftrag.

Die hier gezeigten Komponenten, Zustände und Datenfelder sind ein konzeptioneller Referenzentwurf. Ihre Verbindung wurde nicht implementiert oder experimentell validiert. Standards und Produktdokumentationen tragen einzelne Grundlagen; sie belegen keine getestete Gesamtarchitektur. Die technische Ausarbeitung muss die beschriebenen Grenzen in den tatsächlich gewählten Systemen durchsetzen und nachweisen.

Der praktische Maßstab bleibt die Ausgangsfrage: Kann das Unternehmen erklären und belegen, warum genau diese Handlung unter genau diesen Bedingungen stattfinden durfte? Eine tragfähige Architektur hält dafür den Zusammenhang vom Auftrag bis zur bestätigten Wirkung zusammen. Fehlt eine Voraussetzung, muss sie ebenso nachvollziehbar anhalten können.

Quellen und Entwurfsstatus

Die Quellen sind bei ihren Aussagen im Text verlinkt. Die technische Verbindung der Komponenten, Zustände, Datenfelder und Abnahmevorschläge ist ein eigener konzeptioneller Entwurf. Sie wurde nicht implementiert oder experimentell validiert.

1. SKOS
   https://www.w3.org/TR/skos-reference/
2. OWL
   https://www.w3.org/TR/owl2-overview/
3. SHACL
   https://www.w3.org/TR/shacl/
4. Decision Model and Notation
   https://www.omg.org/spec/DMN/1.5/About-DMN
5. Open Policy Agent
   https://www.openpolicyagent.org/docs
6. Jeff Bezos unterscheidet
   https://d1.awsstatic.com/onedam/marketing-channels/website/aws/en_US/whitepapers/approved/executive-insights/2015-letter-to-shareholders.pdf
7. Palantir-Ontologiearchitektur
   https://www.palantir.com/docs/foundry/ontology/overview
8. AWS-Leitfäden zu agentischer Enterprise-Architektur
   https://docs.aws.amazon.com/prescriptive-guidance/latest/govern-architect-agentic-ai/enterprise-architecture.html
9. Model Context Protocol
   https://modelcontextprotocol.io/specification/2025-11-25/server/tools
10. Temporal-Dokumentation zu Activities
    https://docs.temporal.io/activity-definition

Fassung 4.2 · 2. November 2026. Die Illustrationen zeigen fiktive Orte. Die drei zusätzlichen Schaubilder sind KI-generierte Darstellungen des konzeptionellen Referenzentwurfs. Die ursprünglichen beiden Diagramme liegen als Vektorgrafiken vor.